A Decomposition Approach for Shipboard Manpower Scheduling

Haitao Li

College of Business Administration, University of Missouri – St Louis, St Louis, MO 63121-4400, lihait@umsl.edu

Keith Womer

College of Business Administration, University of Missouri – St Louis, St. Louis, MO 63121-4400, womerk@umsl.edu

ABSTRACT

This paper presents a model and procedure for determining crew composition for a new technology like the Navy's DDX. Here the prototype DDX manpower scheduling problem is modeled as a project scheduling problem with multi-purpose resources (PSMPR) where multi-skilled sailors form teams to accomplish interrelated onboard tasks. A hybrid decomposition algorithm that incorporates constraint programming (CP) and a tabu search (TS) metaheuristic is developed for solving this *NP-hard* problem. In computational experiments, the performance of our hybrid algorithm is tested and compared with solutions found using mixed-integer linear programming (MILP) with CPLEX and with lower bounds obtained from a bin packing problem with conflicts (BPC).

Keywords: manpower scheduling; resource-constrained project scheduling; constraint programming; tabu search; hybrid approach;

INTRODUCTION

The manpower scheduling problem addressed in this paper is motivated by the problem of reducing crew size for the U.S. Navy's DDX class ships. We model this problem by combining resource-constrained project scheduling with generalized assignment models. Our goal in this effort is to develop and explore modeling and computational techniques for this variety of problem. We do not propose to model all of the complexities of the DDX manning problem here though. We believe that with suitable modification such a tool can be used to determine either general crew size and training requirements or specific requirements for a specific mission.

Project scheduling has a long and colorful history in the Navy. The well-known Program Evaluation and Review Technique (PERT) was developed to aid the nuclear submarine development program in the 1950's. There, the optimal sequence of interrelated tasks was determined in order to finish a project or a group of projects by a certain time and within a cost constraint. In the environment where resources can be acquired at will, the optimization involved with project management is now routine. But the resource-constrained project scheduling problem (RCPSP) in the DDX context presents a formidable scheduling problem. Depending on the mission, the list of tasks to be accomplished on board varies from period to period and, in emergency situations, survival may depend on scheduling tasks in the correct order. Furthermore, the jobs must be accomplished with a fixed and multi-skilled crew. A planner has to determine: *i)* starting times of all tasks and time-off (rest) periods; *ii)* an appropriate assignment of sailors to the pool of tasks and skills. Since this problem is related to the project scheduling problem with multi-purpose resources (PSMPR) introduced by Li and Womer (2006), we name it DDX-PSMPR.

As a variant of the standard PSMPR, DDX-PSMPR belongs to the class of assignment-type resource-constrained project scheduling problems (Drexl, Juretzka et al. 1998). It generalizes the single-mode RCPSP (Pritsker, Waters et al. 1969) and is similar to the multi-mode RCPSP (MRCPSP, (Talbot 1982)) in that a task may be performed in more than one way (mode) with different combinations of sailors assigned to the task. The MRCPSP only models the flexibility of executing a task through the trade-off between time and resources, but not the intrinsic ''versatility'' of resource units. As the complexity of tasks' skill requirements and sailors' skill-mix increases, the number of modes in a PSMPR will explode. The PSMPR is also an extension of the classical multi-purpose machine (MPM, (Brucker 2001)) scheduling problems into the more general project scheduling environment by allowing the presence of general temporal constraints.

Modeling the tasks of a DDX crew presents a formidable problem. Briefly, an approach to the problem is as follows. A number of sailors need to be assigned to a group of on-board tasks modeled as a project that will be performed during a scheduling horizon. There are time-related technical requirements among tasks such as precedence relations, minimum/maximum time lags and due dates. A task may require one or multiple skills. We assume that all the relevant skills required by a task must be present simultaneously for the task to progress. A sailor can be described with multiple skills, but can perform only one

APPLICATION AREA:
Manpower & Personnel

OR METHODOLOGIES:
Integer Programming, Metaheuristics

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2009		**00-00-2009 to 00-00-2009**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Decomposition Approach For Shipboard Manpower Scheduling	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Missouri ? St Louis,College of Business Administration,St Louis,MO,63121**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Military Operations Research, V14 N3 2009,Government or Federal Purpose Rights License.

14. ABSTRACT
This paper presents a model and procedure for determining crew composition for a new technology like the Navy's DDX. Here the prototype DDX manpower scheduling problem is modeled as a project scheduling problem with multi-purpose resources (PSMPR) where multi-skilled sailors form teams to accomplish interrelated onboard tasks. A hybrid decomposition algorithm that incorporates constraint programming (CP) and a tabu search (TS) metaheuristic is developed for solving this NP-hard problem. In computational experiments, the performance of our hybrid algorithm is tested and compared with solutions found using mixed-integer linear programming (MILP) with CPLEX and with lower bounds obtained from a bin packing problem with conflicts (BPC).

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**24**	

skill at a time. Sailors must have sufficient rest during their service on board, e.g., at least 8 hours of rest during any 48-hour period. The workload assigned to a sailor during the scheduling horizon cannot exceed the maximum workload capacity of a sailor. There is a deadline for the completion time of all the tasks. The planning decision needs to be made in the intermediate or long run to find the optimal skill-mix of sailors while minimizing the number of sailors needed to man a ship. (Based on the skill mapping solutions, we assume that a training program can be conducted to equip each sailor with the corresponding skills they need.) This requires us to find a schedule with both the starting time of each task and the right sailor-task/skill assignment.

The remainder of the paper is organized as follows. The next section provides some background about the U.S. Navy's DDX project, the on-board operations, and the related past work in manpower planning and scheduling. Next we give a formal description of the DDX manpower scheduling problem with a numeric example and present its mixed-integer linear programming (MILP) formulation. Next, a bin packing model with conflicts (BPC) for the problem with fixed time scheduling of tasks and rest periods is introduced. This is used to obtain a lower bound of the number of sailors required to man a ship. We then present a decomposition heuristic algorithm with the cooperation of constraint programming (CP) and a tabu search (TS) procedure. The next section provides experimental results. Finally we draw conclusions and mention future research opportunities.

BACKGROUND

DDX is the U.S. Navy's future multi-mission surface combatant designed to deliver precision strike and fire support, dominate the littoral environment and defeat the most challenging threats (U.S.Navy 2005). In November 2001, the U.S. Navy announced that it would issue a revised Request for Proposal (RFP) for the Future Surface Combatant Program. The former DD-21 program would now be known as DDX. Two concerns about the old DD-21 program are: first, it was much larger than the current DDG-51 Arleigh Burke-class destroyers; second, the Navy was investing too much in a ship primarily designed to accommodate the long-range Advanced Gun System (AGS). The program focus would now be on a family of advanced technology surface combatants, rather than a single class. In April 2002, Northrop Grumman Ship Systems was selected as the lead design agent for DDX. Currently, DDX has achieved all of its development milestones on schedule. A national team, led by Northrop Grumman Ship Systems (prime contractor) and Raytheon (weapon and electronic systems integrator), is leading the DDX design effort. The lead ship is scheduled for fleet delivery in 2011 and to enter service in 2013 (U.S.Navy 2005).

The DDX is developing state-of-the-art technologies and systems and integrating them into a complete warfare system. Key technologies critical to DDX include: electric drive and integrated power management systems, multi-function and volume search radar suites, the Advanced Gun System, and new hull design emphasizing efficiency at 30-knots sustained speed, mission payload growth capacity and stealth.

To justify the investment of these high-end technologies the optimal utilization of human resources is required. The meaning of ''optimal utilization'' is twofold. First, the crew size is to be reduced. Although the crew size for DDX is yet to be defined, the threshold is set to be 150. This represents major cost saving compared to crew levels of 330 on Spruance class destroyers and 220 on Oliver Hazard Perry frigates. Second, the reduction in crew size will create an increased need for cross-functional training. The idea is to have sailors trained functionally across warfare areas who can then be flexibly employed as the situation demands. This approach will result in a more compact, flexible and versatile crew. In fact, reducing the number of crew members and improving their performance has been listed as one of the eight strategic objectives for the Navy to maintain the human performance and competence while entering the 21st century (U.S.Navy 2005). Therefore, determining the right number of sailors and their skill-mix while satisfying all the on-board temporal and skill constraints becomes critical for this new technology.

Shipboard Operations

Shipboard operations reside in such a complex system that we need to get a clear sense of what types of operations must be considered. If a cruise is regarded as a project then an entire sequence of tasks must be performed to accomplish the project. The tasks are constrained by temporal relations that must hold among at least some of them and by resource requirements.

Tasks. *Tasks* are defined as those individual jobs required to complete an operation; *operations* are specific jobs that must be done to perform a function; and *functions* are general types of work that must be performed to accomplish a *mission*. That is, a mission is at the highest level, while tasks reside at the lowest level to support it. Hence the *work-breakdown-structure* (WBS) approach in project management fits well in modeling the system of shipboard operations. We model a task as an *activity* with three attributes: starting time, ending time and processing time (duration). It is assumed that the duration of each task is known as a constant, while the starting and ending times are decision variables. Some shipboard routine tasks include check-in or check-out, maintenance actions, preventive maintenance, inspections, predeployment preparations, routine watches, shore patrol, etc. Some emergencies and unique situations include fire fighting, damage control, survival, etc. The collection of tasks for any length of scheduling horizon can be thought of as a project.

Temporal Relations. *Temporal relations* are constraints related to time. They specify one aspect of the technical requirements. Technically, tasks are required to overlap with, precede or follow each other. For example, routine watches must *precede* maintenance; an emergency event must *precede* any other ordinary tasks. Simultaneous execution (overlapping) is allowed as long as some time lags between a pair of tasks are satisfied. That is, we consider the *general temporal constraints* (Neumann, Schwindt et al. 2002) which include the precedence constraints prevalent in the machine scheduling environment as a special case. For example, weekly maintenance cannot start until *at least* 18 hours after inspection starts (*minimum time lag*); repairing must start *at most* 6 hours after inspection completes (*maximum time lag*). We may also impose a *due date* on a task, as well as the *deadline* on the total completion time (makespan) of the entire project. For example, damage control must be completed before the fourth day; the entire project must be finished within 7 weeks.

Resource Requirements. *Resource requirements* consist of another aspect of the technical requirements. The skills that are necessary to perform a task are given; what is not given and needs to be determined is which sailors with what skill sets need to be assigned to a task. Also, a task may require multiple skills and we assume that all the skills required by a task must be present simultaneously. For example, a task of ''repairing'' may require three different skills: electronic engineering, mechanical engineering and computer software. Each of the three required skills needs to be assigned with one sailor.

Rest Periods. As in the traditional crew rostering problem (Kohl and Karisch 2004), *time off* (rest) periods have to be assigned to sailors. Under the shipboard environment, it is necessary to allow enough flexibility for rest periods so that they can be scheduled intelligently to achieve a shorter makespan. This is different from most crew rostering problems where the starting and ending times of rest periods are fixed and only assignment decisions are made concerning time off. Also the rest periods need to be distributed evenly and a minimum number of sailors have to be on duty at any time. In the most flexible scenario, such minimum number of required sailors could vary over time, i.e., a different minimum in each time period. Of course special considerations might also dictate additional constraints. For example, we assume that there is a bunk for each sailor but we might also imagine a policy of ''hot bunking'' where there are fewer bunks than sailors and the number of bunks is an additional constraint. This is not included in the current formulation however. In our modeling approach, the rest periods are modeled together with tasks as *activities* with a fixed duration but unknown starting and ending times.

Past Work

The general problem of manpower planning and scheduling consists of determining how to match the available personnel to personnel requirements of an organization. Burgess and Busby (1992) classified this problem into three categories: manpower planning, personnel scheduling and personnel assignments. Manpower planning, according to Gass (1991), determines the number of personnel and their skills that best meets the future operational requirements of an organization. Personnel scheduling usually involves setting work patterns (shifts) and assignment of personnel to these shifts to meet job demands that vary over time. Personnel assignments deal with the assignment of personnel to tasks and work stations. A recent survey by Ernst, Jiang et al. (2004) decomposes personnel scheduling into six sub-modules associated with the process of constructing a roster. The difficulty with the current personnel scheduling and rostering approaches, as pointed out by Ernst, Jiang et al. (2004), is that *"It is usually not computationally practical to deal simultaneously with all the modules required together to generate a roster, though such an approach is desirable from the perspective of creating the best overall rosters."* They also foresee that greater quality gains may be obtained not by improving algorithms for solving any individual module of the problem, but by integrating more modules into a single problem.

The DDX manpower scheduling problem studied in this paper is such an attempt that integrates both scheduling and assignment decisions into a single problem. This has been achieved by applying an assignment-type resource constrained project scheduling model, namely, the project scheduling problem with multi-purpose resources (PSMPR).

PROBLEM DESCRIPTION AND FORMULATION

The DDX-PSMPR can be formally described as follows. A set of on-board tasks J have to be finished by a deadline $\bar{T}$. Each task $j \in J$ has a constant processing time p_j. A set F of general temporal relations exists among tasks. δ_{ij} denotes the minimum time lag between the starting times of task i and j, i.e., task j cannot start until at least δ_{ij} hours after task i starts. A due date d_j is imposed for some task $j \in J$. Each task $j \in J$ requires a set of skills $K^j \subseteq K$, where K is the entire skill set relevant for execution of the project. Without loss of generality, we assume that each skill $k \in K$ requires one sailor to perform it. Should a task require more than one sailor with the same skill then it is modeled as requiring multiple skills. A set S of sailors is available. Each sailor $s \in S$ may be supplied with a maximum of Δ skills and has a maximum workload W that cannot be exceeded during the scheduling horizon. A sailor has to take a rest with a duration of τ hours every λ hours. We further make the following assumptions:

A1) No preemption is allowed, i.e., a task cannot be interrupted once it starts;
A2) The set of skills K^j have to be present simultaneously for task j to progress;
A3) A sailor can perform at most one skill at any time point, i.e., each sailor is treated as a *unary resource*.

The objective is to find a feasible roster of starting times for all the tasks and rest periods and assignment of sailors to tasks and skills (determining the selected sailors' skill-mix) that minimizes the total number of sailors required to man a ship.

A Numeric Example

Table 1 provides an example of the PSMPR problem for DDX manpower scheduling.

Ten tasks need to be scheduled over a three-day horizon, thus the deadline on project makespan is 72 hours. Each task has a constant processing time given in column two. The binary matrix in column three specifies the type of skills (K1, K2, K3, K4) required by each task. For instance, task J1 requires the skill of type 2 (K2). Temporal relations are expressed in the form of $< i, j, \delta_{ij} >$ as shown in column four, where δ_{ij} denotes the minimum time lag between start of task i and the start of j. For example, task J9 cannot start until at least 14 hours after J1 starts (minimum time lag); task J10 must start at most 4 hours after J2 starts (maximum time lag). The last column lists the due date

Table 1. An example of the DDX-PSMPR problem.

Task	Processing times (in hours)	Skill requirements	Temporal relations	Due date (in hours)
J1	13	[0, 1, 0, 0]	< J1, J9, 14 > < J1, J5, 10 >	15
J2	6	[0, 0, 1, 0]	< J2, J10, 3 > < J2, J8, 6 > < J2, J5, 7 >	40
J3	7	[1, 0, 0, 0]	< J3, J4, 9 > < J3, J5, 9 >	20
J4	9	[0, 0, 0, 1]	< J4, J10, 8 > < J4, J7, 6 > < J4, J8, 8 >	30
J5	13	[0, 1, 0, 0]	< J5, J6, 6 >	50
J6	11	[0, 0, 1, 0]	< J6, J1, -19 >	60
J7	8	[1, 0, 0, 0]	< J10, J2, -4 >	50
J8	12	[1, 0, 0, 0]	-	60
J9	13	[0, 0, 1, 0]	-	40
J10	15	[0, 0, 1, 0]	-	50

for each task. Each sailor has a maximum workload of 40 hours which cannot be exceeded over the scheduling horizon and must have a rest period of 8 hours assigned every 48 hours.

Figure 1 provides two feasible solutions to the above example. For each sailor, the assignment of tasks and skills is determined (hence the skills mix of each sailor is determined); the work/rest schedule is also specified by the starting times of tasks and rest periods (rest periods are represented by shaded rectangles). Solution-1 requires 4 sailors whereas Solution-2 requires only 3 sailors. Notice that both solutions satisfy all temporal relations, due dates, skill requirements, workload constraints and the deadline on makespan. Both schedules require sailors to work for long periods of time but we only required one eight hour rest period in 48 hours. Clearly, Solution-2 achieves a better objective value as a result of having a ''smoother'' schedule (with a makespan of 59), whereas Solution-1 depends on a ''tighter'' schedule (with a makespan of 41). That is, the schedule in Solution-2 is a better ''leveled'' schedule compared with Solution-1 over the scheduling horizon of 72 hours. Hence it leads to a resource profile with fewer sailors but requires them for a longer period of time. This also reflects the time-resource trade-off relationship in project management and has the flavor of a resource leveling problem.

When the skill-mix of sailors does not exist, i.e., each sailor can perform only one skill, the DDX-PSMPR reduces to one type of resource leveling problem, namely, the resource investment problem (RIP, (Neumann, Schwindt et al. 2002)), which is *NP-hard* (Neumann and Zimmermann 1999). The DDX-PSMPR itself is distinct from the RIP due to multi-skilled personnel, which requires that assignment decisions be made in addition to scheduling decisions. Most of the existing resource leveling methods, both exact (see e.g., Mohring (1984) and Demeulemeester (1995)) and heuristic (see e.g., Woodworth and Willie (1975), Ahuja (1976), Harris (1990), Savin, Alkass et al. (1997), and Brinkmann and Neumann (1996)) procedures, assume that there is no explicit resource constraint. Neumann and Zimmermann (1999) present priority-rule based heuristics for resource leveling problem with a wide variety of objective functions, general temporal relations and explicit resource constraints. New solution approaches need to be developed for the assignment-type resource leveling problems such as the DDX-PSMPR. In the following sections, we first present the MILP formulation of the DDX-PSMPR, which proves to be intractable for problems of reasonable size. We then introduce a three-phase decomposition heuristic algorithm that utilizes a combination of methods to solve the problem both efficiently and effectively.

MILP Formulation of the DDX-PSMPR

The maximum workload W of a sailor is set to be $\max\{W_0, \theta \cdot \bar{T}\}$, where θ is a parameter

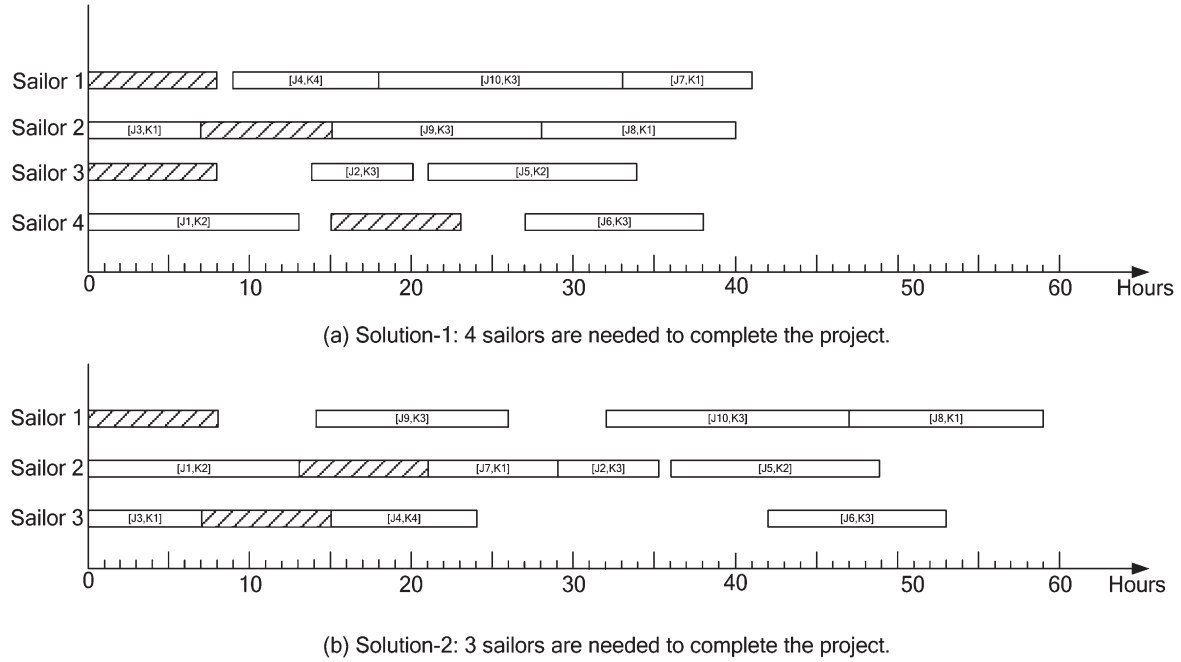

Figure 1. Gantt charts of two feasible solutions to the example in Table 1. The shaded bars represent required rest periods.

representing the percentage of working time during the scheduling horizon. For example, if we assume that a sailor's workload is 8 hours every 24 hours, then θ is set be 1/3. W_0 is a lower-threshold for W. Let M be a large positive integer, l is the index of time periods ($l \in L = \{1, \ldots, \bar{L}\}$), where $\bar{L}$ is calculated by $\lceil \bar{T} / \lambda \rceil$. Then the MILP formulation of the problem can be stated as follows:

Sets and Parameters

J : set of tasks
K : set of skills
S : set of sailors
F : set of temporal relations
K^j : set of skills required by task j
$\bar{T}$: deadline on all tasks in hours
p_j : processing time (duration) of task j in hours
δ_{ij} : minimum time lag between task i and task j in hours
d_j : due date of task j
λ : length of a time interval in hours
τ : length of rest period in hours
$\bar{L}$: number of time intervals during the scheduling horizon
θ : workload factor as a percentage of the total number of hours during the scheduling horizon
W : workload limit of each sailor
Δ : maximum number of skills assigned to a sailor
W_0 : lower threshold for a sailor's workload during the scheduling horizon
M : large integer

Decision variables

z_s : 1 if and only if sailor s is selected to man the ship; 0 otherwise.
x_{jks} : 1 if and only if sailor s is assigned to skill k in task j ; 0 otherwise.
φ_{ks} : 1 if and only if sailor s is assigned to skill k ; 0 otherwise.
t_j : starting time of task j.
t_{sl} : starting time of the l-th rest period of sailor s.
y_{ij} : 1 if and only if task i precedes task j, 0 otherwise.
y_{sl}^j : 1 if and only if task j precedes rest period of sailor s in period l, 0 otherwise.
y_j^{sl} : 1 if and only if rest period of sailor s in period l precedes task j, 0 otherwise.

Objective function

$$\min \sum_{s \in S} z_s$$

Constraints

$$\sum_{s \in S} x_{jks} = 1 \quad \forall j \in J, k \in K^j \tag{1}$$

$$\sum_{j \in J} \sum_{k \in K^j} p_j x_{jks} \le W \cdot z_s \quad \forall s \in S \tag{2}$$

$$\varphi_{ks} \ge x_{jks} \quad \forall j \in J, k \in K^j, s \in S \tag{3}$$

$$\sum_{k \in K} \varphi_{ks} \le \Delta \cdot z_s \quad \forall s \in S \tag{4}$$

$$x_{jks} + x_{jk's} \le 1 \quad \forall j \in J; k, k' \in K^j, k \ne k', s \in S \tag{5}$$

$$t_i - t_j \ge \delta_{ij} \quad \forall (i,j) \in F \tag{6}$$

$$t_j + p_j \le \min\{d_j, \bar{T}\} \quad \forall j \in J \tag{7}$$

$$t_{sl} \ge (l-1) \cdot \lambda \quad \forall s \in S, l \in L \tag{8}$$

$$t_{sl} + \tau \le l \cdot \lambda \quad \forall s \in S, l \in L \tag{9}$$

$$t_j \ge t_i + p_i - M \cdot (1 - y_{ij}) \quad \forall (i,j) \in J \times J, i \ne j \tag{10}$$

$$t_{sl} \ge t_j + p_j - M \cdot (1 - y_{sl}^j) \quad \forall j \in J, s \in S, l \in L \tag{11}$$

$$t_j \ge t_{sl} + \tau - M \cdot (1 - y_j^{sl}) \quad \forall j \in J, s \in S, l \in L \tag{12}$$

$$y_{ij} + y_{ji} \ge x_{iks} + x_{jk's} - 1$$
$$\forall (i,j) \in J \times J, i > j; k \in K^i, k' \in K^j; \forall s \in S \tag{13}$$

$$y_{sl}^j + y_j^{sl} \ge x_{jks} \quad \forall j \in J, k \in K^j, s \in S, l \in L \tag{14}$$

$$t_j, t_{sl} \ge 0$$

$$z_s, x_{jks}, \varphi_{ks}, y_{ij}, y_{sl}^j, y_j^{sl} \in \{0,1\}$$

Constraints (1) through (5) take care of the assignment aspect of the problem. The remaining Constraints (6) through (14) formulate the scheduling aspect of the problem. Constraint (1) assigns exactly one sailor to each skill required by a task. Constraint (2) enforces that the maximum workload of the selected sailors cannot be exceeded. Constraint (3) states that a sailor must be assigned to a certain skill to be able to work on the same skill in any tasks. Constraint (4) enforces the maximum number of skills assigned to a sailor. Constraint (5) prevents a sailor from being assigned to skills required by the same task due to assumption *A2)* and *A3)*. Constraint (6) states the generalized precedence relations and (7) enforces the due dates for tasks; together they guarantee time feasibility of the task schedule. When δ_{ij} is positive, (6) specifies a minimum time lag between i and j ; when δ_{ij} is negative, (6) becomes a maximum time lag between j and i ; when δ_{ij} equals the processing time p_j of activity j, (6) reduces to a precedence relationship specifying that j precedes i. Constraints (8) and (9) restrict the time frame for each rest period. Constraints (10) through (12) are the *big-M formulation* in the classical *disjunctive programming approach*. Constraints (13) and (14) state that the sequence of two activities (tasks or rest periods) has to be determined if they involve the same sailor due to the *unary* resource assumption *A3)*.

As problem size increases, the size of the integer programming model explodes as shown in Figure 2. Increasing problem size from 10 tasks and 4 skills to 60 tasks and 8 skills will increase the number of variables by a factor of 20 and the number of constraints by a factor of 200. Although some state-of-the-art software with advanced presolving techniques, such as CPLEX, is able to reduce the model size significantly, a real world problem with reasonable size can easily involve hundreds of thousands of variables and millions of constraints, which would be a formidable task for the traditional exact methods such as branch-and-bound or branch-and-cut to solve.

LOWER BOUND BASED ON A BIN PACKING MODEL

The DDX-PSMPR includes two ingredients: a scheduling and an assignment problem. If we are given a feasible schedule for the project, time

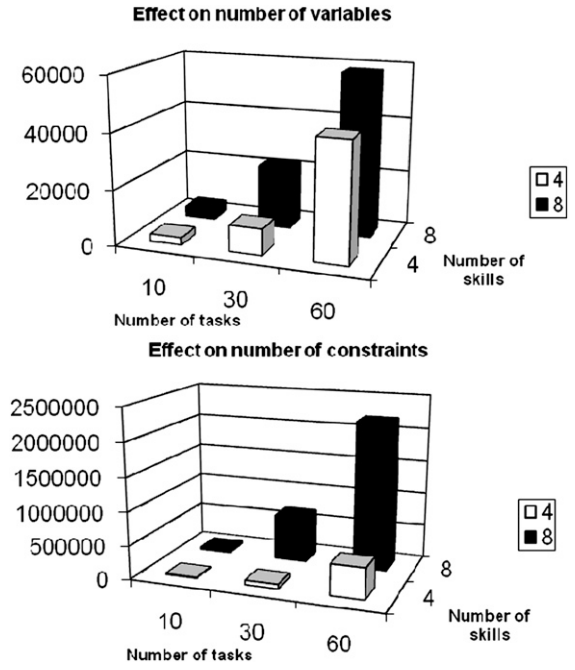

Figure 2. The effect of problem parameters on MILP model size.

slots with starting and ending times for all tasks and rest periods will be fixed. Then the problem of finding an assignment with the minimum number of sailors to these time slots can be modeled as a bin packing problem with conflicts (BPC, (Jasen 1999)). Such a transformation is used to obtain a lower bound for the number of sailors needed to man the ship without resort to the schedule of tasks. It also provides insights for the hybrid decomposition algorithm that will be developed on the next page.

Sailor Assignment BPC Given a Fixed Time Schedule

Let O be the set of all task-skill pairs and H be the set of rest periods. We define an activity set $V = O \cup H$ that includes all task-skill pairs and rest periods of duration w_1, w_2,…,$w_{|V|}$. Given a fixed time schedule, the starting times of all activities are determined, i.e., $t_i = \bar{t}_i (i = 1, \ldots, |V|)$. Then the sailor assignment BPC can be defined as follows. Two activities are *conflicting* if they cannot be assigned to the same sailor. The problem is to assign activities to the least number of sailors while ensuring that the total weights (workloads) of all tasks assigned a sailor does not exceed W and no sailors can be assigned to *conflicting* activities. Let graph $G(V, E)$ be the sailor assignment *conflict graph* with vertex set V and edge set E. An edge (i, j) exists if activities i and j are *conflicting* with each other. A pair of activities (i, j) are *conflicting* when one of the following holds:

i) $w_1 + w_2 > W$.
ii) Activities i and j represent skills required by the same task, hence need to be present simultaneously (Assumption *A2)*).
iii) Activities i and j overlap in the given schedule, i.e.,

$$\max\{\bar{t}_i + w_i, \bar{t}_j + w_j\} - \min\{\bar{t}_i, \bar{t}_j\} < w_i + w_j. \tag{15}$$

ii) and *iii)* are due to *Assumption A3)* that a sailor is treated as a unary resource.

The integer linear program for the sailor assignment BPC can be written as:

$$\min \sum_{s \in S} z_s$$

subject to

Constraints (1) through (4)

$$x_{iks} + x_{jk's} \leq 1 \quad \forall (i,j) \in E, k \in K^i, k' \in K^j, s \in S \tag{16}$$

When one of the *conflicting* pair i represents a rest period, (16) becomes (17):

$$x_{jks} = 0 \quad \forall (i,j) \in E, k \in K^j, \tag{17}$$

which prevents assigning s to any skills required by j. Constraints (16) are not needed when both i and j represent rest periods since no assignment for a rest period is needed (a rest period is always associated with a specific sailor). The following two observations can be made concerning the BPC model:

O1) The conflict graph $G(V,E)$ can be constructed by any feasible schedule without knowing the assignment solutions to the DDX-PSMPR.
O2) The restrictiveness of the BPC model is directly affected by the number of Constraints (16), i.e., the density of the conflict graph $G(V,E)$. The sparser is $G(V,E)$,

potentially the better objective value (fewer number of sailors) can be achieved.

The above BPC model is used in two different ways in our decomposition algorithm. Prior to the main solving phase, it is used to obtain a lower bound to the original problem; during the algorithm iterations, its feasibility version is used to solve the Phase-III assignment subproblem to extend the partial schedule to a complete solution. The BPC model for finding the lower bound (BPC-LB) is obtained by constructing the edge set E checking only conditions *i)* and *ii)*. The feasibility version of the BPC model for finding a feasible assignment to the original problem (BPC-FEAS) can be obtained by constructing E checking all three conditions.

Obtaining a Lower Bound

We obtain a lower bound to the DDX-PSMPR by solving the BPC-LB model. However, a BPC is *NP-hard* as it directly generalizes the bin-packing problem with no conflicts (BPP) which is known to be *NP-hard* (Garey and Johnson 1979). Hence our focus is to obtain a lower bound to the BPC. Two lower bounds for the BPC have been studied by Gendreau, Laporte et al. (2004): $L_1 = \lceil \sum_{i=1}^{|V|} w_i / W \rceil$ is the simple lower bound to the BPP; while the so called constrained packing lower bound also takes conflict constraints into account. Their computational study indicates that the quality of the two lower bounds depends largely on the density of the conflict graph G. Specifically, when the density value is below 30% they are almost identical. Since the conflict graph associated with our DDX sailor assignment BPC is rather sparse (see computational results in Table 5), L_1 would be a reasonable choice. Martello and Toth (1990) have shown that L_1 has a worst-case performance of 1/2 and only performs well when sizes of items w_i (durations of activities in our case) are sufficiently small with respect to the workload capacity W, which may not be true in real-world problems. Hence instead of using L_1, we use the better lower bound L_2 in Martello and Toth (1990), which is proved to have a worst-case performance of 2/3 and always dominates L_1 . L_2 *is* obtained by differentiating items with size larger and smaller than half of the bin capacity. We refer to Martello and Toth (1990) for a detailed description of the procedure to obtain L_2 with a time complexity of $O(n)$.

A HYBRID DECOMPOSITION ALGORITHM

Many recent studies have shown the efficiency and effectiveness of hybrid constraint programming (CP) and mixed-integer linear programming (MILP) approaches for solving challenging combinatorial optimization problems (see, for example, Easton, Nemhauser et al. (2004), Benoist, Laburthe et al. (2001), Eremin and Wallace (2001), Jain and Grossmann (2001), Chu and Xia (2005), Timpe (2002), and Hooker (2004) among others). CP is generally good at solving scheduling problems due to its effectiveness in handling binary constraints through arc consistency algorithms (Hooker 2002). In addition, the descriptive nature of CP often makes a model compact and flexible, hence considerably reduces the size of a model and makes the model easy to modify in a changing real-world environment. The drawback of CP, however, is its lack of relaxation techniques or bounds which are well developed in the OR community (Hooker 2002). It is often costly for CP alone to explore the entire solution space in a naïve fashion of gradually tightening the objective function (for a general introduction of constraint propagation and search algorithms used in CP, we refer to Marriott and Stuckey (1998)). On the other hand, the MILP approach often depends on disjunctive programming formulation for modeling scheduling constraints, which results in a large number of sequencing variables and big-M formulations (as in Constraints (10) through (12)) that make the relaxation less effective. Hence our core consideration in designing hybrid approach for solving the DDX-PSMPR is to separate the scheduling aspect and assignment aspect of the problem and use CP to handle the scheduling sub-problem, which is otherwise difficult for MILP to handle.

The three-phase hybrid algorithm for solving the DDX-PSMPR can be sketched as follows. Given a lower bound $\underline{N}$ of the number of sailors (which is L_2 obtained in the previous section), Phase-I of the procedure models and solves a single-mode RCPSP feasibility problem (RCPSP-FEAS) by CP methods to obtain

a feasible schedule Ω_{CP}. Then Ω_{CP} is used to construct the corresponding *conflict graph* $G(V, E)$, according to Observation *O1)*. Phase-II solves a resource leveling problem with general temporal relations and explicit resource constraints to reduce the density of $G(V, E)$ with the objective function of minimizing the number of overlapping activities in the schedule. It "smoothes" the schedule Ω_{CP} obtained in Phase-I to get a less sparse conflict graph $G'(V, E)$, which leads to a less restrictive BPC problem to be solved in Phase-III (Observation *O2)*) and potentially a better objective value to the original problem. Due to the *NP-hardness* of the Phase-II resource leveling problem itself, we design a tabu search (TS) metaheuristic to get a well-smoothed schedule within reasonable computational time. Finally, Phase-III solves a feasibility version of the BPC problem (BPC-FEAS described in the previous section) based on $G'(V, E)$ to obtain a complete feasible solution to the original DDX-PSMPR problem.

Phase-I Scheduling Subproblem

Define Sailors as a *discrete resource* with capacity N by treating the N sailors homogeneously. Now the Phase-I single-resource RCPSP-FEAS CP model written in OPL, a programming language that supports both mathematical programming and constraint programming ((Van Hentenryck 1999) and (ILOG 2005)), is described as follows:

RCPSP-FEAS

{

$$\text{activity}[i].start - \delta_{ij} \geq \text{activity}[j].start \quad \forall (i,j) \in F \tag{18}$$

$$\text{activity}[j].start + p_j \leq \min\{d_j, \bar{T}\} \quad \forall j \in J \tag{19}$$

$$\text{activity}[j] \ \textit{requires} \ (|K^j|) \ \text{Sailors} \quad \forall j \in J \tag{20}$$

$$N_{\text{iter}} := N_{\text{iter}-1} + 1 \quad (\text{iter} \geq 1) \tag{21}$$

$$N_0 = L_2 \tag{22}$$

}

The italicized words in (18) through (22) are keywords in OPL (except the notations for sets and parameters). Constraints (18) and (19) take care of the general temporal relations as in Constraints (6) and (7). Constraints (20) specify the resource constraints, i.e., task j requires $|K^j|$ units of the *discrete resource* Sailors. Notice that no sequencing variables or big-M formulations are needed in the CP formulation, thus the model size is considerably reduced. RCPSP-FEAS is a relaxed version of the scheduling side of the original DDX-PSMPR in the following ways: *i)* the set of sailors is treated homogeneously as a single *discrete resource* prior to being assigned with skills; *ii)* an activity is defined to be associated with each task instead of each required skill. It follows that an activity has only one way (mode) to be executed, thus reducing the original scheduling problem into a single-mode RCPSP. Furthermore, RCPSP-FEAS obtains a feasible solution instead of an optimal one, which further reduces the computational effort. Next we state the following property.

Property 1. If RCPSP-FEAS is infeasible, the original DDX-PSMPR is infeasible.

Proof. It follows from the fact that RCPSP-FEAS is a relaxed problem of DDX-PSMPR. Q.E.D.□

Infeasibility of the RCPSP-FEAS indicates that no feasible solution to the original problem exists at the current capacity N. Then one needs to add a "cut" that increases N to resolve infeasibility. This is achieved by Constraint (21), which increments the capacity of Sailors by one at iteration i over the previous iteration $i-1$ of the procedure. It excludes all solutions with N or less than N sailors. Constraint (22) initiates N_0 to be L_2.

Phase-II Resource Leveling Procedure

Let Ω be any feasible schedule to RCPSP-FEAS. The Phase-II procedure solves a resource leveling problem minimizing the number of overlapping tasks in Ω. Let $A^{\Omega}(t)$ be the set of activities in progress at time t, also called *active set*, i.e.,

$$A^{\Omega}(t) := \{ j \in V | t_j \leq t < t_j + p_j \} \quad \forall t \in \{0, \ldots, \bar{T}\} \tag{23}$$

Then the number $r^{\Omega}(t)$ of Sailors used at time t can be obtained as:

$$r^{\Omega}(t) := \sum_{j \in A^{\Omega}(t)} |K^j| \quad \forall t \in \{0, \ldots, \bar{T}\} \tag{24}$$

We define χ_{ij} as a binary variable which equals 1 if activities i and j overlap and 0 otherwise. We also define end_{ij} as the maximum end time of i and j; $start_{ij}$ as the minimum start time of i and j. The Phase-II resource leveling problem can be described as follows:

$$\min \quad f_{\text{overlap}}(\Omega) = \sum_{i \in V} \sum_{j \in V, j > i} \chi_{ij}$$

subject to:

Temporal constraints (4) through (7)

$$start_{ij} \le t_i \quad \forall (i,j) \in V \times V, j > i \tag{25}$$

$$start_{ij} \le t_j \quad \forall (i,j) \in V \times V, j > i \tag{26}$$

$$end_{ij} \ge t_i + w_i \quad \forall (i,j) \in V \times V, j > i \tag{27}$$

$$end_{ij} \ge t_j + w_j \quad \forall (i,j) \in V \times V, j > i \tag{28}$$

$$M \cdot \chi_{ij} \ge w_i + w_j - (end_{ij} - start_{ij}) \quad \forall (i,j) \in V \times V, j > i \tag{29}$$

$$r^{\Omega}(t) \le N_{\text{iter}} \quad \forall t \in \{0, \ldots, \bar{T}\} \tag{30}$$

$$t_i \ge 0, \ \chi_{ij} \in \{0,1\}$$

If the right-hand side of (29) is positive, then χ_{ij} will be forced to be 1. Constraints (30) describe the resource constraints, i.e., the total resource utilization $r^{\Omega}(t)$ at time t of the *discrete resource* Sailors cannot exceed its capacity N_{iter} at iteration *iter*. Following Baker (1974), an objective function is called *regular* if $f' > f$ implies $t_j' > t_j$ for some $j \in V$. Clearly, the objective function $f_{\text{overlap}}(\Omega)$ studied here belongs to the class of *nonregular* objective functions as delaying an activity does not necessary increase the objective function value. Most of the existing priority rule based heuristics deal with regular objective functions; non-regular objective functions often require more than a simple heuristic. Thus we propose a tabu search procedure for this resource leveling problem with $f_{\text{overlap}}(\Omega)$. For basic concepts of tabu search, we refer to Glover and Laguna (1997). It is important to point out that solving the Phase-I RCPSP-FEAS sub-problem has served as the construction phase in tabu search. That is, the non-smoothed feasible schedule Ω_{CP} from Phase-I is used as the initial solution for the TS improvement procedure.

The Neighborhood. Various types of neighborhoods have been proposed in the project scheduling literature (see Neumann, Schwindt et al. (2003) for an updated survey). Most of them, e.g., activity list based (Baar, Brucker et al. (1997), Bouleimen and Lecocq (1998) and Cho and Kim (1997)), random key based (Cho and Kim (1997), Naphade, Wu et al. (1997)), schedule scheme based (Baar, Brucker et al. 1997) neighborhoods, deal with project scheduling problems minimizing project makespan (*regular* objective function) with only precedence relationships among activities. The neighborhoods that can handle both *nonregular* objective functions and general temporal constraints include the simple starting time vector (Icmeli and Erenguc 1994), restricted starting time vector (Neumann and Zimmermann 2000) and order-based neighborhoods (Neumann, Schwindt et al. 2003). For simplicity, we name them *N*1, *N*2 and *N*3, respectively. Neighbors in *N*1 are obtained by shifting some activity by one time unit within its earliest-latest starting time window. Possible violations of precedence constraints are penalized by a penalty function. The neighborhood structure in *N*2 is defined by the so called *decision set*, which contains the "locally optimal" start times for certain objective functions (Neumann, Schwindt et al. 2002). The cardinality of a decision set is usually linear in the number of activities, restricting the number of time points to explore. The time feasibility of a neighboring schedule is preserved by updating the starting times of other activities appropriately. Possible violations of resource constraints are allowed by imposing a penalty cost function. *N*3 is based on the fact that each feasible schedule can be associated with some strict order in the activity set. *N*3 consists of sets of spanning forests and spanning trees of so called *order-based networks* exploiting the order-based structure of the feasible region. *N*2 is advantageous over *N*1 in that the neighborhood size is considerably smaller without sacrificing effectiveness. The implementation of *N*3 requires

complex data structures and to our knowledge no empirical study has been reported about the quality of *N*3. Hence we adapt *N*2 to cope with the resource leveling problem minimizing the number of overlapping activities ($f_{\text{overlap}}(\Omega)$).

Consider an unscheduled activity j with earliest start time $ES_j = 1$, latest start time $LS_j = 8$ and processing time $p_j = 2$ in a partial schedule. The decision set D_j for $f_{\text{overlap}}(\Omega)$ can be illustrated by Figure 3. Two consecutive *jump discontinuities* are $t_1 = 2$ and $t_2 = 5$ (a *jump discontinuity* is a time point when at least one activity is started or at least one activity is completed). The decision set D_j is given by $\{t_1, t_2, ES_j, LS_j\}$, where each decision point is marked by a $\times$ on the Time axis. Instead of exploring all the time points $\{1,\ldots,8\}$ in the time window of j, we consider only the four time points in the decision set. It is straightforward to observe that there exists a local minimizer t_2 with a penalty of 1 (activity j only overlaps with activity b causing the number of overlapping activities to increase by 1).

For detailed mathematical description and proof of the decision set employed in our approach, we refer to Neumann and Zimmermann (1999).

Procedures for Implementing Tabu Search. Tabu search introduced by Glover (1989a, 1989b) is a local search metaheuristic to avoid local optima in solving combinatorial optimization problems. It repeatedly moves from the current solution Ω to a schedule Ω' with the best objective value in the neighborhood of Ω denoted by $N(\Omega)$. An important consideration in TS is to prevent "cycling", i.e., prevent revisiting a solution recently visited. This is achieved by making a move "tabu" for a predefined number of iterations called *tabu list size* (*TL*), which is also known as *short-term memory*. If a new best solution is found, the tabu status of the corresponding move is canceled (*aspiration criterion*). Next we describe the main procedures for implementing a simple TS that utilizes only the *short-term memory* component.

Given the current schedule Ω, its neighbor schedule Ω' is obtained by replacing the starting time t_j of any activity $j \in V$ by a time point t in D_j. For simplicity, we use $m(t_j \leftarrow t)$ to represent this operation. A move $m(t_j \leftarrow t)$ is a *valid move* if it satisfies the following conditions: *i)* $t \in D_j$; *ii)* m cancels at least one pair of overlapping activities; *iii)* m is resource feasible. Condition *ii)* further reduces the number of neighbors to be considered. Condition *iii)* avoids the use of a penalty cost function for possible violations of resource constraints; it also reduces the number of schedules to scan by excluding resource infeasible moves.

We choose *NI* as the maximum number of iterations without improving the best objective value f^*_{overlap} found so far. The TS algorithm for the Phase-II resource leveling problem with f_{overlap} is summarized in Figure 4.

Simply executing a move $m(t_j \leftarrow t)$ does not guarantee time feasibility to the temporal constraints. Define *Reach(j)* as the set of nodes in the project network that can reach node i and *Reachable(j)* as the set of nodes reachable from node j. We then follow the approach in Neumann and Zimmermann (1999) to keep time feasibility of the schedule by making induced moves after $m(t_j \leftarrow t)$ as described in Figure 5.

We let $f(m)$ be the objective value of a move m and f^{move} denote the best move value. Figure 6

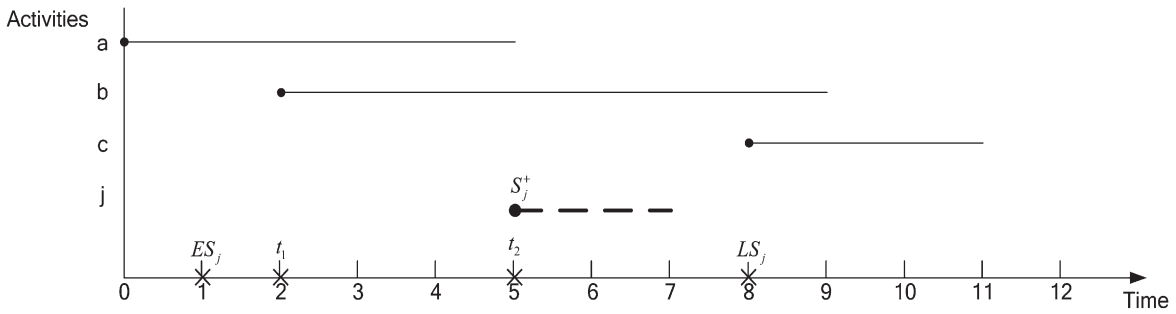

Figure 3. The neighborhood defined by D_j of activity j.

Algorithm TS: A TS Algorithm for resource leveling with f_{overlap}

Step 1.

Set $\Omega^* := \Omega_{CP}$, $\Omega := \Omega_{CP}$ and $f^*_{\text{overlap}} := +\infty$

Step 2.

While $iter < NI$

Call ***Find_Best_Move(*** Ω ***)***

Execute $m^*(t_j \leftarrow t)$

Call ***Make_Time_Feasible(*** $m^*(t_j \leftarrow t)$ ***)*** to get Ω'

Update the tabu list

If $f(m^*) < f^*_{\text{overlap}}$, **then**

Set $\Omega^* := \Omega'$ and $f^*_{\text{overlap}} := f(m^*)$

Set $iter := 0$

Set $iter := iter + 1$

End Algorithm

Figure 4. Algorithm TS.

summarizes the procedure for finding the best move m^* in the current neighborhood. We use the standard *aspiration criterion* that if a move improves the current best solution, its tabu status (if any) will be canceled. We explore the entire set of valid moves using *first-improve-strategy* (FIS). That is, whenever an improvement over the current objective value f_{overlap} (Ω) is found the neighborhood search is terminated and the improvement move is set to be the best move m^*. FIS helps to speed up the search, especially at the early stage of the algorithm.

The Decomposition Algorithm

Figure 7 depicts the framework of our three-phase decomposition heuristic for solving the DDX-PSMPR.

The algorithm starts by obtaining a lower bound L_2 on the number of sailors needed. L_2 is then used to initialize the capacity N_0 of the discrete resource Sailors in the Phase-I RCPSP-FEAS sub-problem to start the procedure. To resolve infeasibility of the Phase-I sub-problem, a ''cut'' (we call it Cut-1) is generated by increasing N_{iter} by one, i.e., setting $N_{\text{iter}} := N_{\text{iter}} + 1$ to exclude all solutions with N_{iter} sailors. Cut-1 is a valid cut, as infeasibility of RCPSP-FEAS proves that the original DDX-RCPSP is not feasible under the current availability of N_i sailors (Property 1). This process repeats until a feasible schedule Ω_{CP} is found to the RCPSP-FEAS. Recall that the activities in our model include both real tasks and rest periods. Increasing the number of sailors not only increments the resource capacity, but also creates new rest periods associated with the newly added sailor(s) to schedule. Hence, we always obtain a different schedule from Phase-I, which prevents cycling, i.e., feeding the same schedule to the Phase-III BPC-FEAS problem. Our computational study has indicated no such problem of cycling for the tested problems.

Procedure 1: ***Make_Time_Feasible(*** $m(t_j \leftarrow t)$ ***)***

If $t > t_j$, **then**

For $i \in$ *Reach(j)* **do** $t_i := \max(t_i, t + d_{ji})$

Else If $t < t_j$, **then**

For $h \in$ *Reachable(j)* **do** $t_h := \min(t_h, t - d_{hj})$

End Procedure

Figure 5. Procedure 1 to repair a schedule to be time feasible.

The procedure then proceeds with the Phase-II resource leveling procedure minimizing the number of overlapping activities (Algorithm TS) to get a smoothed schedule Ω_{TS} with fewer number of overlapping activities for constructing the Phase-III BPC-FEAS sub-problem. The Phase-II procedure further makes the partial schedule different from the one in the previous iteration.

Infeasibility of BPC-FEAS indicates that N_{iter} sailors are not sufficient to support Ω_{TS}. This is due to the workload and maximum number of skills constraints, which have not been considered in the scheduling phases (Phase-I and Phase-II) of the decomposition procedure. Hence another cut (Cut-2) is added by increasing N_i by one and a new iteration starts. Cut-2, however, may or may not be a valid cut since there may exist some schedule other than Ω_{TS} under the same current capacity N_{iter} that could make the Phase-III sub-problem feasible. Hence our decomposition algorithm may or may not achieve optimality as long as Cut-2 is generated. If only Cut-1 is generated during the procedure, however, it is straightforward to prove that the solution found by our algorithm is optimal by Property 1. The key component in this decomposition algorithm is the Phase-II resource leveling procedure. It produces a smoother schedule, thus a less restrictive BPC to be solved in Phase-III.

Procedure 2: ***Find_Best_Move(*** Ω ***)***

Step 1.

Determine resource utilization vector $R(\Omega)$ and set of discontinuity time points $D(R)$

Set $f^{move} := +\infty$ and $TF := \max_{j \in V}(LS_j - ES_j)$

Step 2.

For $t = 0$ to TF

  For $j = 0$ to $|V|$

    If $m(t_j \leftarrow t)$ is a valid move, **then**

    Execute $m(t_j \leftarrow t)$

    Call ***Make_Time_Feasible(*** $m(t_j \leftarrow t)$ **)**

    If $m(t_j \leftarrow t)$ has no tabu status **or** $f(m) < f^*_{\text{overlap}}$ found so far (*aspiration criterion*), **then**

      If $f(m) < f^{move}$, **then**

        Set $m^* := m(t_j \leftarrow t)$ and $f^{move} := f(m)$

        If $f(m^*) < f_{\text{overlap}}(\Omega)$, **then** terminate (*first-improve-strategy*)

End Procedure

Figure 6. Procedure 2 to find the best move in the neighborhood.

COMPUTATIONAL EXPERIMENTS

Since currently there is no benchmark for the addressed problem and the sensitive data from the Navy could not be used directly, we generate problem instances for algorithm testing purposes. As the DDX-PSMPR shares common data concerning the project network with the resource constrained project scheduling problem (RCPSP), we adapt the project scheduling part of the RCPSP instances for the DDX-PSMPR. A well-known problem generator of RCPSP is ProGen, which was used to generate the RCPSP benchmark bank PSPLIB (Kolisch, Sprecher et al. 1995). However, ProGen generates RCPSP problems with only precedence constraints. Hence we use ProGen/max (Schwindt 1996) that is able to generate RCPSP with *general temporal constraints*: minimal and maximal time lags.

In order to evaluate the performance of our decomposition algorithm compared with CPLEX, we have concentrated on the problem space where CPLEX had a chance to find quality solutions. We control two factors affecting the problem size of DDX-PSMPR: the number of tasks $|J|$ and number of skills $|K|$ in the project. Let $|J| \in \{10, 30, 60\}$ and $|K| \in \{4, 8\}$. We also control another important factor, the deadline $\bar{T}$ on the project makespan. We follow Drexl and Kimms (2001) to control $\bar{T}$:

$$\bar{T} = DF \cdot \max_{j \in J} EF_j \qquad \forall j \in J, \tag{31}$$

where EF_j is the earliest finishing time of task j. We set $DF = 1.25$ for the scenario with a ''tight'' deadline and allow a 50% longer deadline for the ''loose'' deadline scenario. That is, we allow DF vary in the set $\{1.25, 1.875\}$. There are $3 \times 2 \times 2 = 12$ combinations for the three factors. For each combination we generate 5 instances, i.e., $5 \times 12 = 60$ test instances in total. In our experiments, the maximum number of skills assigned to each sailor Δ is set to be 7. Table 2 summarizes the main information about these instances.

The Phase-I scheduling sub-problem is modeled and solved using ILOG Solver 6.0 (ILOG 2003a) and ILOG Scheduler 6.0 (ILOG 2003d) by constraint programming. The Phase-II tabu search procedure is coded in C++ and

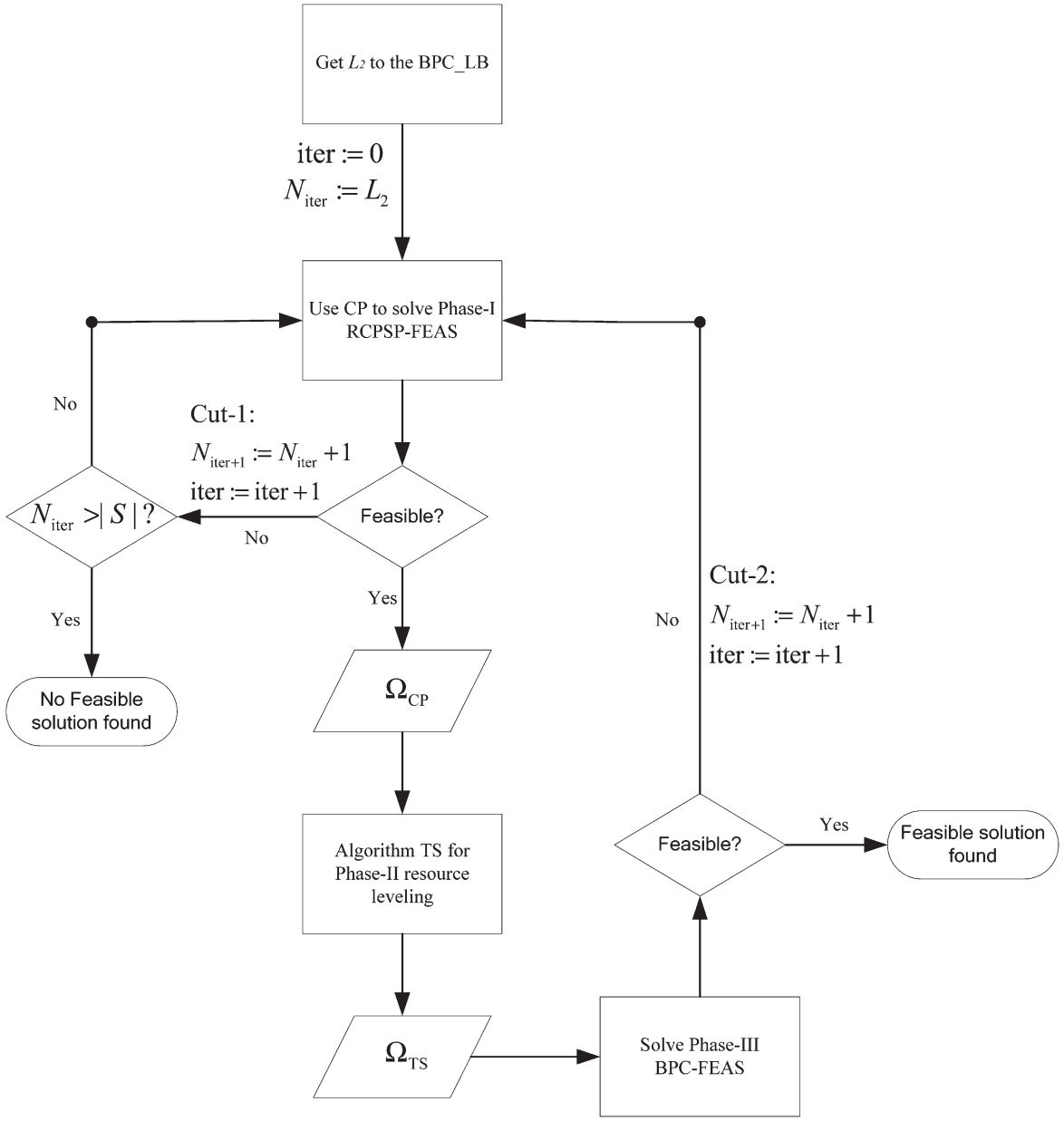

Figure 7. Flow chart of the three-phase hybrid decomposition algorithm.

connected with other phases through the C++ API provided by ILOG Concert Technology 2.0 (ILOG 2003b). The Phase-III BPC-FEAS integer program is solved by ILOG CPLEX 9.0 (ILOG 2003c). All computations are performed on Pentium IV PC with 2.8 GHz CPU speed and 512Mb RAM. A running time limit of 10 seconds is imposed for Phase-I and Phase-III each. We set $NI = 20$ for the tabu search procedure. Our computational experience indicates that a small tabu list size TL often enables the TS procedure to improve the objective function fast. A good candidate for TL is 2.

Table 2. Summary of the Test Instances. $W = \max\{W_0, \theta \cdot \bar{T}\}$, where $W_0 = 40$.

				Loose deadline DF = 1.875		Tight deadline DF = 1.25	
Instance name	|J|	|K|	|S|	$\bar{T}$(hrs)	W(hrs)	$\bar{T}$(hrs)	W(hrs)
DDX1_10T_4S	10	4	20	75	40	50	40
DDX2_10T_4S	10	4	20	75	40	50	40
DDX3_10T_4S	10	4	20	75	40	50	40
DDX4_10T_4S	10	4	20	75	40	50	40
DDX5_10T_4S	10	4	20	75	40	50	40
DDX1_10T_8S	10	8	30	75	40	50	40
DDX2_10T_8S	10	8	30	75	40	50	40
DDX3_10T_8S	10	8	30	75	40	50	40
DDX4_10T_8S	10	8	30	75	40	50	40
DDX5_10T_8S	10	8	30	75	40	50	40
DDX1_30T_4S	30	4	30	180	64	120	40
DDX2_30T_4S	30	4	30	180	64	120	40
DDX3_30T_4S	30	4	30	210	72	140	48
DDX4_30T_4S	30	4	30	180	64	120	40
DDX5_30T_4S	30	4	30	270	96	180	64
DDX1_30T_8S	30	8	50	180	64	120	40
DDX2_30T_8S	30	8	50	180	64	120	40
DDX3_30T_8S	30	8	50	210	72	140	48
DDX4_30T_8S	30	8	50	180	64	120	40
DDX5_30T_8S	30	8	50	270	96	180	64
DDX1_60T_4S	60	4	50	360	120	240	80
DDX2_60T_4S	60	4	50	360	120	240	80
DDX3_60T_4S	60	4	50	480	160	320	112
DDX4_60T_4S	60	4	50	330	112	220	80
DDX5_60T_4S	60	4	50	300	104	200	72
DDX1_60T_8S	60	8	60	360	120	240	80
DDX2_60T_8S	60	8	60	360	120	240	80
DDX3_60T_8S	60	8	60	480	160	320	112
DDX4_60T_8S	60	8	60	330	112	220	80
DDX5_60T_8S	60	8	60	300	104	200	72

Model and Algorithm Performance

Table 3 provides the computational results on the 60 test instances. We report the objective value (Obj), number of iterations (Iter) of the decomposition algorithm and running time (CPU in seconds) for finding the best solutions. In order to show the important role of the Phase-II tabu search for resource leveling, we record the results of both the three-phase hybrid decomposition algorithm with tabu search (we call it Three-Phase) and the two-phase hybrid algorithm without tabu search (we call it Two-Phase). An underlined Obj value indicates that Three-Phase finds a better solution than Two-Phase does. Clearly, Three-Phase finds more better solutions when the deadline is loose. This is because a loose deadline leaves more ''room'' for the resource leveling procedure in Phase-II to reduce the number of overlapping tasks, hence achieving more density reduction of the corresponding conflict graph. As the deadline becomes tight, however, not much resource leveling can be achieved in Phase-II, thus the performance of Three-Phase converges to Two-Phase.

As for computational efficiency, we observe that Three-Phase always requires fewer iterations than Two-Phase to find equal or better quality solutions for the tested instances. This is because the resource leveling phase makes the Phase-III assignment problem easier to

Table 3. Computational results of the Three-Phase and Two-Phase Decomposition Algorithms.

	Loose deadline DF = 1.875						Tight deadline DF = 1.25					
	Three-Phase			Two-Phase			Three-Phase			Two-Phase		
Instance name	Obj	Iter	CPU(s)	Obj	Iter	CPU(s)	Obj	Iter	CPU(s)	Obj	Iter	CPU(s)
DDX1_10T_4S	5	2	2.38	5	2	0.20	5	1	0.13	5	1	0.11
DDX2_10T_4S	5	2	2.20	5	2	0.16	5	2	0.61	5	2	0.16
DDX3_10T_4S	4	2	1.72	7	5	0.64	5	2	0.97	7	4	0.43
DDX4_10T_4S	5	2	2.83	5	2	0.49	5	2	0.66	5	2	0.05
DDX5_10T_4S	4	2	1.52	5	3	0.36	4	2	0.67	5	3	0.38
DDX1_10T_8S	9	2	5.44	10	4	13.16	11	3	22.42	11	3	21.69
DDX2_10T_8S	10	3	23.64	11	4	38.30	11	4	30.99	11	4	38.31
DDX3_10T_8S	10	2	11.78	13	5	41.19	11	4	33.41	13	5	41.75
DDX4_10T_8S	11	3	22.84	12	4	30.23	12	4	31.01	12	4	30.19
DDX5_10T_8S	13	5	45.44	15	7	60.34	15	7	30.91	15	7	30.20
DDX1_30T_4S	7	2	52.29	9	4	30.22	11	2	28.83	11	2	10.16
DDX2_30T_4S	7	2	30.36	7	2	10.13	11	2	22.94	11	2	10.17
DDX3_30T_4S	8	3	105.56	9	4	30.33	11	2	24.59	11	2	10.16
DDX4_30T_4S	7	2	33.61	7	2	10.13	11	3	51.90	11	3	20.22
DDX5_30T_4S	7	3	79.95	7	3	21.22	9	3	50.58	9	3	20.25
DDX1_30T_8S	21	6	248.20	23	7	62.64	32	5	120.92	32	5	42.34
DDX2_30T_8S	20	5	182.26	20	5	41.77	32	5	102.26	32	5	42.38
DDX3_30T_8S	18	4	153.39	18	4	31.34	27	6	146.37	27	6	52.23
DDX4_30T_8S	17	4	116.21	17	4	31.04	25	3	78.53	27	5	41.84
DDX5_30T_8S	14	4	205.42	17	7	62.39	22	6	181.75	23	7	62.41
DDX1_60T_4S	9	4	699.98	9	4	30.99	11	2	115.98	11	2	10.53
DDX2_60T_4S	9	3	755.36	12	6	51.91	13	3	369.31	13	3	20.86
DDX3_60T_4S	7	3	557.14	7	3	23.94	10	3	373.84	10	3	20.83
DDX4_60T_4S	10	4	577.62	10	4	31.02	12	2	90.62	12	2	10.56
DDX5_60T_4S	11	3	751.08	12	4	31.27	14	2	166.10	15	3	20.97
DDX1_60T_8S	18	4	1719.86	21	7	68.31	26	4	503.50	26	4	34.31
DDX2_60T_8S	23	7	3103.61	23	7	69.75	30	6	1364.35	30	6	58.22
DDX3_60T_8S	16	4	1723.85	21	10	114.11	24	8	1947.94	27	10	115.83
DDX4_60T_8S	21	5	1185.54	23	7	68.14	29	6	668.67	31	6	57.03
DDX5_60T_8S	22	4	923.43	22	7	34.89	32	6	627.00	32	6	58.20

1. *Obj*: objective value, number of sailors. *Iter*: number of iterations finding the solution. *CPU(s)*: algorithm running time in seconds.
2. Underlined values indicate that the Three-Phase algorithm finds a better solution than the Two-Phase algorithm.

solve. Not surprisingly, Three-Phase generally spends more computational time than Two-Phase on average due to the additional resource leveling phase. Given the nature of the scheduling problem considered in this paper, one would probably favor solution quality more than speed. Interestingly, for problems with 10 tasks, 8 skills and a loose deadline, Three-Phase requires less overall computational time while finding better solutions. This is because the size of these problem instances is so small that each run of the Phase-II tabu search procedure spends less than 1 second, which is trivial compared to the overall computational time.

Figure 8 shows the average percentage of reduction in objective value by the Three-Phase algorithm over the Two-Phase algorithm when $|J|$ and $|K|$ vary. The Three-Phase algorithm dominates Two-Phase in the entire problem space defined by $|J|$ and $|K|$, due to the effective role of the Phase-II resource-leveling. On average, the tabu search procedure in Phase-II is able to eliminate 35% of the overlapping activities, i.e., to reduce the density of the conflict graph

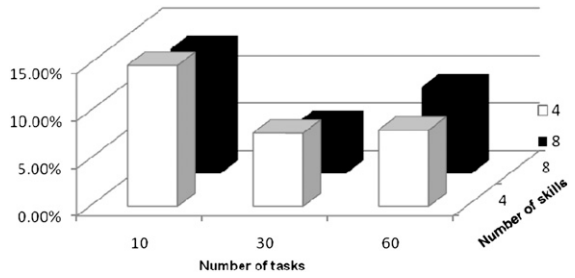

Figure 8. Average percentage of reduction in objective value by Three-Phase over Two-Phase when |J| and |K| vary.

$G(V,E)$ by 35%. It turns out that the conflict graph $G'(V,E)$ associated with the leveled schedule is quite sparse with an average density of 10%, which justifies our choice of the lower bound L_2.

Parameter effects on the behavior of the DDX-PSMPR model are summarized by Table 4. As we can see, the number of skills $|K|$ has the greatest impact on the objective value to the DDX-PSMPR. When $|K|$ increases from 4 to 8, the average number of sailors needed is more than doubled (from 8 to 19). This is mainly due to the fact that the average number of skills required by each task increases as $|K|$ increases. Notice that the computational effort, represented by the average number of iterations and running time, also increase significantly as $|K|$ increases. That is, a DDX-PSMPR problem with a large $|K|$ is often difficult to solve. Hence an appropriate skill aggregation is critical in deploying the model in practice.

The second most influential factor is the deadline factor DF. The average number of sailors needed increases 33% (from 12 to 16) when the deadline of project becomes 50% tighter (DF from 1.875 to 1.25). This reflects the well-known time-resource/cost trade-off in project management. A DDX-PSMPR problem with a tighter deadline (smaller DF) is usually more difficult to solve as more iterations are needed for finding the best solution. However, this does not necessarily mean that more computational time is needed. Notice that the average running time for $DF = 1.25$ is only 54% of that for $DF = 1.875$, although an average of one more iteration is needed for $DF = 1.25$. In other words, the three-phase decomposition algorithm spends more time per iteration for problems with a loose deadline. This is because the Phase-II tabu search procedure is able to run longer given the same parameter NI for problems with a loose deadline ($DF = 1.875$). And more density reduction on the conflict graph can be achieved for problems with $DF = 1.875$, which is supported by our computational experience: about 22% overlapping is eliminated for problems with $DF = 1.25$, in contrast to almost 50% overlapping eliminated for problems with $DF = 1.875$.

Now we turn our attention to the effect of the number of tasks $|J|$. Interestingly, the objective value of the DDX-PSMPR problem is not significantly driven by $|J|$. When $|J|$ increases, $\bar{T}$ usually increases too, making the workload of each sailor increase proportionally (see Table 2). That is, when the number of items increases the capacity of each bin increases simultaneously in a bin packing problem, which offsets the effect of each other. (We do observe a big change in objective value when $|J|$ increases from 10 to 30. It is because for $|J| = 10$ the workload is set to be the threshold $W_0 = 40$ instead of following the proportional relationship to $\bar{T}$). What drives the number of needed sailors is not necessarily the number of tasks in the project, but the complexity of skill requirement ($|K|$) and how constrained the project network is ($\bar{T}$). This is a desirable behavior of our DDX-PSMPR model. One often needs to schedule projects with different scheduling horizon in

Table 4. Effects of parameters on the model and Three-Phase algorithm performance.

	|J|			|K|		DF	
Performance	10	30	60	4	8	1.875	1.25
Avg. Obj	8	16	17	8	19	12	16
Avg. Iter	3	4	4	2	5	3	4
Avg. CPU(s)	13.58	100.79	916.03	165.04	521.89	444.15	242.78

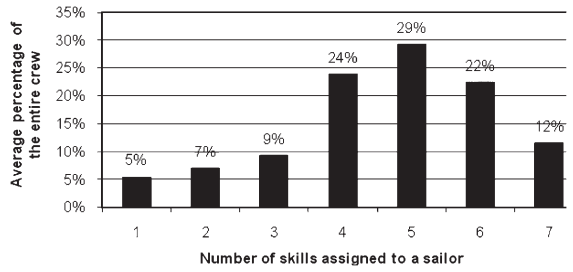

Figure 9. Average crew composition of problem instances with 60 tasks, 8 skills and maximum number of assigned skills $\Delta = 7$.

practice and it is possible for the same number of sailors to perform projects with different lengths. Furthermore, we notice that the number of iterations does not increase significantly as $|J|$ increases, which indicates that having more tasks in the project does not necessarily make the problem more difficult for our decomposition algorithm to solve. However, the running time needed for finding the best solution does increase. This is because more computational effort is needed to obtain a smoothed schedule in Phase-II due to a larger neighborhood size.

We also record the number of skills assigned to each sailor for problem instances with 60 tasks and 8 skills. Figure 9 shows the average percentage of each category of sailors classified by the number of assigned skills. Sailors with five skills consist of the largest portion (29%) of the crew. Only a smaller portion of the crew is required to possess extremely large or small number of skills, which seems to be a desirable behavior of the model.

Comparison with Lower Bound and CPLEX

Table 5 presents the computational results of the three-phase hybrid decomposition algorithm compared with the lower bound L_2 and CPLEX. L_2 may or may not be feasible as it is a lower bound to the BPC-LB discussed earlier. CPLEX is used to solve the MILP model at equations 1–14. We have chosen a CPLEX parameter setting that balances feasibility and optimality. A running time limit of 50 hours is imposed for CPLEX on each instance. The CPLEX solutions reported in Table 5 are the best solutions found by CPLEX within the time limit. An empty entry indicates that CPLEX could not find a feasible solution within the time limit or ran out of memory. Numbers in bold font indicate the solutions to the corresponding problem instance are optimal. As CPLEX may spend a long time trying to prove optimality before terminating, to make a fair comparison we report the time CPLEX spends reaching an optimal solution instead of the total running time for proving optimality.

We evaluate the performance of Three-Phase and CPLEX on the basis of the following criteria:

i) the percentage of instances for which a feasible solution is found (ρ_{feas}),
ii) the percentage of instances for which an optimal solution is found and optimality is proven (ρ_{opt}),
iii) the mean percentage deviation dev_{LB} of the objective value found from lower bound LB, and
iv) the mean computation time for finding the best solution in seconds (t_{cpu}).

While ρ_{feas} and ρ_{opt} refer to all instances, dev_{LB} and t_{cpu} correspond only to the instances for which a feasible solution is found by the respective approach.

At first, we compare the overall performance of our Three-Phase algorithm and CPLEX in Table 6. Three-Phase is able to solve all 60 problem instances, whereas CPLEX fails to find feasible solutions for 21 instances. Most of the unsolved instances by CPLEX are large in size. For instance, a typical instance with 60 tasks and 8 skills results in an MILP model of 50,000 variables (columns) and 2,000,000 constraints (rows), which is formidable for CPLEX to handle. As for the algorithm effectiveness, Three-Phase has an average gap from the lower bound of less than 10% and reaches optimality for 20 instances. (One can verify that Three-Phase also proves optimality for these 20 instances as the lower bound is equal to the objective value of the solution found by Three-Phase.) CPLEX on average has a higher gap and is only slightly better in finding and proving optimality for 21 instances. As for the algorithm efficiency, Three-Phase spends less than 7 minutes averagely for finding the best solution for an instance, while it takes CPLEX averagely 5.5 hours, which

Table 5. Comparison with lower bound and CPLEX.

	Loose deadline DF = 1.875					Tight deadline DF = 1.25			
	CPLEX			Three-Phase		CPLEX		Three-Phase	
Instance name	L_2	OBJ	CPU(s)	OBJ	CPU(s)	L_2	OBJ	OBJ	CPU(s)
CPU(s)									
DDX1_10T_4S	5	**5**	3.42	**5**	2.38	5	**5**	89.22	**5**	0.13
DDX2_10T_4S	5	**5**	5.53	**5**	2.20	5	**5**	90.84	**5**	0.61
DDX3_10T_4S	4	**4**	458.93	**4**	1.72	4	5	65.42	5	0.97
DDX4_10T_4S	5	**5**	143.29	**5**	2.83	5	**5**	98.49	**5**	0.66
DDX5_10T_4S	4	**4**	1.89	**4**	1.52	4	**4**	25.14	**4**	0.67
DDX1_10T_8S	9	**9**	1207.90	**9**	5.44	9	11	21236.30	11	22.42
DDX2_10T_8S	9	10	445.69	10	23.64	9	10	6647.56	11	30.99
DDX3_10T_8S	10	**10**	1622.04	**10**	11.78	10	**10**	2157.72	11	33.41
DDX4_10T_8S	10	12	2185.83	11	22.84	10	11	6660.44	12	31.02
DDX5_10T_8S	10	11	1924.32	13	45.44	10	**13**	20.56	15	30.91
DDX1_30T_4S	7	**7**	47330.20	**7**	52.29	11	**11**	8196.03	**11**	28.83
DDX2_30T_4S	7	**7**	3521.73	**7**	30.36	11	**11**	5288.33	**11**	22.94
DDX3_30T_4S	7	**7**	4398.56	8	105.56	11	**11**	8243.53	**11**	24.59
DDX4_30T_4S	7	**7**	73912.70	**7**	33.61	10	11	2140.55	11	51.90
DDX5_30T_4S	6	**6**	46758.10	7	79.95	8	**8**	21986.80	9	50.58
DDX1_30T_8S	18	-	-	21	248.20	29	-	-	32	120.92
DDX2_30T_8S	18	-	-	20	182.26	29	38	33420.30	32	102.16
DDX3_30T_8S	16	-	-	18	153.39	23	29	27176.60	27	146.37
DDX4_30T_8S	15	-	-	17	116.21	24	39	59316.10	25	78.53
DDX5_30T_8S	12	26	44129.00	14	205.42	18	26	103997.00	22	181.75
DDX1_60T_4S	7	-	-	9	699.98	11	15	75260.40	**11**	115.74
DDX2_60T_4S	8	-	-	9	755.36	12	21	58725.20	13	369.31
DDX3_60T_4S	6	-	-	7	557.14	9	-	-	10	373.84
DDX4_60T_4S	8	-	-	10	577.75	12	14	97347.00	**12**	90.62
DDX5_60T_4S	10	-	-	11	751.18	14	17	26064.80	**14**	166.10
DDX1_60T_8S	16	-	-	18	1719.86	24	-	-	26	503.50
DDX2_60T_8S	18	-	-	23	3103.61	27	-	-	30	1364.35
DDX3_60T_8S	14	-	-	16	1723.85	19	-	-	24	2043.80
DDX4_60T_8S	18	-	-	21	1186.18	25	-	-	29	668.67
DDX5_60T_8S	20	-	-	22	923.92	28	-	-	32	627.00

is nearly 60 times the amount of time required by Three-Phase.

Next, we compare the performance of Three-Phase and CPLEX in different problem spaces. Table 7 compares the above performance criteria given different number of tasks. Clearly, the ability of CPLEX to find feasible solutions drops significantly as the number of tasks increases. CPLEX's solution quality also deteriorates as reflected by the drastic increase in dev_{LB}. To be specific, CPLEX has a slight advantage in dev_{LB} when the number of task is small (10 tasks), then CPLEX is 50% higher in dev_{LB} when |J| increases to 30, and is almost three times as high as the dev_{LB} of Three-Phase when |J| reaches

Table 6. Comparison of overall performance of Three-Phase and CPLEX.

	Three-Phase	CPLEX
ρ_{feas}	100.00%	65.00%
ρ_{opt}	33.33%	35.00%
dev_{LB}	9.16%	11.04%
t_{cpu}	343.47	20316.17

Table 7. Comparison of performance when |J| varies.

	10		30		60	
	Three-Phase	CPLEX	Three-Phase	CPLEX	Three-Phase	CPLEX
ρ_{feas}	100.00%	100.00%	100.00%	75.00%	100.00%	20.00%
ρ_{opt}	55.00%	60.00%	30.00%	45.00%	15.00%	0.00%
dev_{LB}	7.88%	6.05%	8.21%	12.50%	11.40%	30.47%
t_{cpu}	13.58	2254.53	100.79	32654.37	916.03	64356.10

60. CPLEX also spends a considerably large amount of time (almost 18 hours) averagely for finding feasible solutions to large instances with 60 tasks. But realize that in reality, an onboard project can easily include hundreds of tasks which will be formidable for CPLEX to handle.

Table 8 compares the algorithm performance when the number of skills changes. Again Three-Phase shows advantage over CPLEX in both solution quality and speed. Also notice that the mean deviation from the lower bound increases when |K| increases. The reason for this is probably that the presence of more skills results in a larger density of the conflict graph, hence the quality of the lower bound L_2 decreases. In this case, the quality of the lower bound can be improved by replacing L_2 with the constrained packing lower bound (Gendreau, Laporte et al. 2004).

We now turn to the effect of DF on the algorithm performance presented in Table 9. Interestingly, CPLEX finds fewer feasible solutions to problems with a loose deadline than those with a tight deadline. For those solved problems, though, CPLEX finds solutions with better quality. In contrast, Three-Phase appears to be more robust in solution quality.

In order to explore the performance of our decomposition algorithm on larger problems, we add additional 2 GB physical memory to the testing PC (from 512 Mb to 2512 Mb) and were able to solve problems of larger size. Table 10 shows the computational results on five large instances. None of these instances were able to be handled by CPLEX. The memory was exhausted even before the model was extracted by CPLEX, not to mention the computational process.

Our computational experience indicates that memory may become an issue for solving problems with reasonably large size. As we attempted to further increase the problem size, both decomposition algorithms ran out of memory after several iterations. This difficulty can certainly be avoided by using super computers to perform the computation.

CONCLUSIONS AND FUTURE RESEARCH

In this paper, we have formulated and studied the prototype of the U.S. Navy's DDX manpower scheduling problem modeled as a project scheduling problem with multi-purpose resources (PSMPR). It attempts to find not only the optimal schedule of on-board operations but also the optimal assignment (skill-mix) of sailors, providing a new approach for

Table 8. Comparison of performance when |K| varies.

	4		8	
	Three-Phase	CPLEX	Three-Phase	CPLEX
ρ_{feas}	100.00%	80.00%	100.00%	50.00%
ρ_{opt}	60.00%	60.00%	6.67%	10.00%
dev_{LB}	5.38%	6.38%	12.95%	18.48%
t_{cpu}	165.04	20007.63	521.89	20809.82

Table 9. Comparison of performance when DF varies.

	1.875		1.25	
	Three-Phase	CPLEX	Three-Phase	CPLEX
ρ_{feas}	100.00%	53.33%	100.00%	76.67%
ρ_{opt}	33.33%	40.00%	33.33%	30.00%
dev_{LB}	9.19%	6.29%	9.13%	14.34%
t_{cpu}	444.15	14253.07	242.78	24533.97

Table 10. Computational results on five large instances solved by the Three-Phase heuristic.

|J|	|K|	$\bar{T}$(hrs)	W(hrs)	LB	OBJ	CPU(s)
90	8	350	100	30	35	2667.14
90	8	350	100	33	39	3046.89
120	8	520	100	41	45	1345.72
120	8	520	100	40	43	1029.66
120	8	520	80	50	53	940.92

determining the personnel composition in a both time and resource constrained scheduling environment. Unlike the traditional crew rostering models, our model treats the time-off periods as activities in a project and determines their optimal number and starting times flexibly. An advantage of this modeling approach is its ability to incorporate both scheduling and rostering problems into one problem while considering multi-skilled personnel.

We also present a hybrid decomposition heuristic algorithm that incorporates constraint programming (CP) for handling the scheduling part of the problem, and a simple tabu search (TS) that utilizes only short-term memory for the resource leveling phase. Test instances were generated to evaluate the algorithm performance. Our algorithm is also compared with CPLEX and lower bounds obtained from a bin packing with conflicts (BPC) model. Experimental results show that our algorithm is able to find feasible solutions to all test instances and achieves satisfactory solution quality while spending only 1.6% of the computational time required by CPLEX on average. As the problem size increases, the advantage of our algorithm over CPLEX increases in both solution quality and speed. These indicate that our algorithm is highly promising in dealing with large size problems in the real world.

Our current model does not capture all the complexities of the DDX problem. Many side constraints need to be considered when dealing with the real world scenario. One may consider constraints concerning sailors' welfare and preferences as often encountered in personnel scheduling and planning. For example, rest periods have to be distributed evenly during the planning horizon; a sailor prefers not working during weekends and so on. While not easily modeled by integer programming, these side constraints can be elegantly expressed by CP. Once the relevant data become available, our goal is to obtain the realistic version of the DDX manning problem and to deploy this decision tool for the Navy.

From an algorithmic perceptive, there is still room for improving the solution quality by tuning the Phase II TS resource leveling procedure. For example, the tabu list size may be tuned for problems with different size; intensification and diversification that utilize intermediate and long term memory may be added into the simple TS framework for some difficult problem instances. It will also be interesting to seek ways to reduce the memory requirement of the decomposition algorithm, which is especially important for solving large size problems.

ACKNOWLEDGEMENTS

This work was supported by Dr. Janet Spoonamore under the auspices of the U.S. Army Research Office Scientific Services Program administered by Battelle (Delivery Order 0355, Contract No. W911NF-07-D-0001), and the Office of Naval Research (ONR) under Grant No. N00140310621. We would also like to thank the anonymous referees for their helpful and constructive comments.

REFERENCES

Ahuja, H. N. 1976. *Construction Performance Control by Networks.* New York, Wiley.

Baar, T., P. Brucker and S. Knust. 1997. Tabu-search algorithms for the resource-constrained project scheduling problem. *Technical Report*, University Osnabruck.

Baker, K. R. 1974. *Introduction to Sequencing and Scheduling.* New York, Wiley.

Benoist, T., F. Laburthe and B. Rotternbourg. 2001. *Lagrange relaxation and constraint programming collaborative schemes for traveling tournament problems.* Integration of AI and OR Techniques in Constraint.

Bouleimen, K. and H. Lecocq. 1998. A new efficient simulated annealing algorithm for the resource constrained project scheduling

problem. *European Journal of Operational Research* **149**(2): 268–281.

Brinkmann, K. and K. Neumann. 1996. Heuristic procedures for resource-constrained project scheduling with minimal and maximal time lags: the resource-levelling and minimum project duration problems. *Journal of Decision Systems* **5**: 129–155.

Brucker, P. 2001. *Scheduling Algorithms.* Berlin, Springer-Verlag.

Burgess, W. J. and R. E. Busby. 1992. Personnel Scheduling. *Handbook of Industrial Engineering.* G. Galvendy. New York, John Wiley**:** 2155–2169.

Cho, J. H. and Y. D. Kim. 1997. A simulated annealing algorithm for resource constrained project scheduling problems. *Journal of the Operational Research Society* **48**: 735–744.

Chu, Y. and Q. Xia. 2005. *A hybrid algorithm for a class of resource-constrained scheduling problems.* Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2005), Springer.

Demeulemeester, E. 1995. Minimizing resource-availability costs in time-limited project networks. *Management Science* **41**: 1590–1598.

Drexl, A., J. Juretzka, F. Salewski and A. Schirmer. 1998. New modeling concepts and their impact on resource-constrained project scheduling. *Project Scheduling: Recent Models, Algorithms and Applications.* J. Weglarz, Kluwer Academic Publishers.

Drexl, A. and A. Kimms. 2001. Optimization guided lower and upper bounds for the resource investment problem. *Journal of Operational Research Society* **52**: 340–351.

Easton, K., G. Nemhauser and M. Trick. 2004. CP based branch-and-price. *Constraint and integer programming.* M. Milano, Springer. **27**.

Eremin, A. and M. Wallace. 2001. Hybrid Benders decomposition algorithms in constraint logic programming. *LNCS* (2239): 1–15.

Ernst, A. T., H. Jiang, M. Krishnamoorthy and D. Sier. 2004. Staff scheduling and rostering: a review of applications, methods and models. *European Journal of Operational Research* **153**: 3–27.

Garey, M. R. and D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco, W.H. Freeman.

Gass, S. I. 1991. Military manpower planning models. *Computers & Operations Research* **18**(1): 65–73.

Gendreau, M., G. Laporte and F. Semet. 2004. Heuristics and lower bounds for the bin packing problem with conflicts. *Computers and Operations Research* **31**: 347–358.

Glover, F. 1989a. Tabu search - Part I. *ORSA Journal on Computing* **1**: 190–206.

Glover, F. 1989b. Tabu search - Part II. *ORSA Journal on Computing* **2**: 4–32.

Glover, F. and M. Laguna. 1997. *Tabu Search.* Boston, Kluwer Academic Publishers.

Harris, R. 1990. Packing method for resource levelling (Pack). *Journal of Construction Engineering and Management* **106**(2): 331–350.

Hooker, J. 2002. Logic, optimization and constraint programming. *INFORMS Journal on Computing* **14**(4): 285–321.

Hooker, J. N. 2004. *A hybrid method for planning and scheduling.* Principles and Practice of Constraint Programming (CP2004), Springer.

Icmeli, O. and S. Erenguc. 1994. A tabu search procedure for the resource constrained project scheduling problem with discounted cash flows. *Computers and Operations Research* **21**(8): 841–853.

ILOG 2003a. *ILOG Solver 6.0 User's Manual*, ILOG, Inc.

ILOG 2003b. *ILOG Concert Technology 2.0*, ILOG, Inc.

ILOG 2003c. *ILOG CPLEX 9.0 User's Manual*, ILOG, Inc.

ILOG 2003d. *ILOG Scheduler 6.0 User's Manual*, ILOG, Inc.

ILOG 2005. *OPL Studio 3.7.1 User's Manual*, ILOG Inc.

Jain, V. and I. Grossmann. 2001. Algorithms for Hybrid MILP/CP Models for a Class of Optimization Problems. *INFORMS Journal on Computing* **13**(4): 258–276.

Jasen, K. 1999. An approximation scheme for bin packing with conflicts. *Journal of Combinatorial Optimization* **3**: 363–77.

Kohl, N. and S. Karisch. 2004. Airline crew rostering: problem types, modeling, and

optimization. *Annals of Operations Research* **127**: 223–257.

Kolisch, R., A. Sprecher and A. Drexel. 1995. Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science* **41**(10): 1693–1703.

Li, H. and K. Womer. 2006. Project scheduling with multi-purpose resources: a combined MILP/CP decomposition approach. *International Journal of Operations and Quantitative Management* **12**(4): 305–325.

Marriott, K. and P. J. Stuckey. 1998. *Programming with Constraints.* Cambridge, Massachusetts, The MIT Press.

Martello, S. and P. Toth. 1990. Lower bounds and reduction procedures for the bin packing problem. *Discrete Applied Mathematics* **28**: 59–70.

Mohring, R. H. 1984. Minimizing costs of resource requirements in project networks subject to a fixed completion time. *Operations Research* **32**: 89–120.

Naphade, K. S., S. D. Wu and R. H. Storer. 1997. Problem space search algorithms for resource-constrained project scheduling. *Annals of Operations Research* **70**: 307–326.

Neumann, K., C. Schwindt and J. Zimmermann. 2002. *Project Scheduling with Time Windows and Scarce Resources: Temporal and resource-constrained project scheduling with regular and nonregular objective functions.* New York, Springer.

Neumann, K., C. Schwindt and J. Zimmermann. 2003. Order-based neighborhoods for project scheduling with nonregular objective functions. *European Journal of Operational Research* **149**(2): 325–343.

Neumann, K. and J. Zimmermann. 1999. Resource levelling for projects with schedule-dependent time windows. *European Journal of Operational Research* **117**: 591–605.

Neumann, K. and J. Zimmermann. 2000. Procedures for resource levelling and net present value problems in project scheduling with general temporal and resource constraints. *European Journal of Operational Research* **127**: 425–443.

Pritsker, A. A., L. J. Waters and P. M. Wolfe. 1969. Multi-project scheduling with limited resources: A zero-one programming approach. *Management Science* **16**: 93–108.

Savin, D., S. Alkass and P. Fazio. 1997. A procedure for calculating the weighted-matrix of a neuronal network for resource leveling. *Advances in Engineering Software* **28**: 277–283.

Schwindt, C. 1996. Generation of resource-constrained project scheduling problems with minimal and maximal time lags. *Technical Report WIOR-489*, Institute fur Wirtschafts-theorie und Operations Research, University of Karlsruhe.

Talbot, F. B. 1982. Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. *Management Science* **28**(10): 1197–1210.

Timpe, C. 2002. Solving planning and scheduling problems with combined integer and constraint programming. *OR Spectrum* **24**: 431–448.

U.S.Navy. 2005. "Fact Sheets." from http://peoships.crane.navy.mil/factsheets/ddxfactsheet.htm.

U.S.Navy. 2005. Technology for the United States Navy and Marine Corps, 2000-2035: Becoming a 21st-Century Force, U.S.Navy. Volume 4: Human Resources.

Van Hentenryck, P. 1999. *The OPL Optimization Programming Language.* Cambridge, MIT Press.

Woodworth, B. M. and C. J. Willie. 1975. A heuristic algorithm for resource leveling in multi-project, multi-resource scheduling. *Decision Sciences* **6**: 525–540.